IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#32
LDJ
1-30-04

Applicant(s):	Marc Tremblay and William N. Joy		
Title:	IMPLICITLY DERIVED REGISTER SPECIFIERS IN A PROCESSOR		
Application No.:	09/204,479	Filed:	December 3, 1998
Examiner:	David Y. Eng	Group Art Unit:	2155
Atty. Docket No.:	004-3289		

January 20, 2004

Mail Stop - Appeal Briefs - Patents
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF (37 C.F.R. § 1.192)

This brief is in furtherance of the Notice of Appeal, filed on September 17, 2003. The fees required under 37 C.F.R. § 1.17(c) are provided in the accompanying Transmittal. A petition for a two month extension of time under 37 C.F.R. § 1.136(a) and the required fee under 37 C.F.R. § 1.17(a) are also included. This brief is being transmitted in triplicate pursuant to 37 C.F.R. § 1.192(a).

REAL PARTY IN INTEREST

The real party in interest in this appeal is Sun Microsystems, Inc., as evidenced by the assignment recorded at Reel 9622/Frame 0284.

RELATED APPEALS AND INTERFERENCES

Appellants have no knowledge of any related appeals or interferences.

STATUS OF CLAIMS

Claims 1, 3-17, 19-21, 23 and 24 are presented herein on appeal. Claims 2, 18 and 22 were cancelled by the Applicants. Claims 1, 3-17, 19-21, 23 and 24 were rejected in a final Office action dated June 5, 2003. That final rejection is now appealed.

Claims 1, 3-17, 19-21, 23 and 24 presented herein on appeal are reproduced in the Appendix attached hereto.

STATUS OF AMENDMENTS

An amendment after final is being filed concurrently with this brief. The amendment corrects minor grammatical errors in the claims, and does not touch the merits of the claims. The amendments are believed to place the claims in better condition for appeal. As of the mailing of this appeal, these amendments to the claims have not been entered.

SUMMARY OF INVENTION

This presently claimed invention relates to instruction execution by a processor, and in some realizations, to instruction execution elements of a Very Long Instruction Word (VLIW) processor including control elements that define and supply register specifiers. A functional unit executes an instruction that operates on multiple registers in a register file. At least one of the registers operated on by the instruction is identified explicitly by a register specifier that is part of the instruction. At least one additional register operated on by the instruction is identified implicitly, based on the explicitly identified register specified as part of the instruction.

Referring to the discussion of Figure 6 beginning on page 17 of the specification, for example, a processor according to one form of the invention may support a multiply and add instruction encoded as follows:

muladd rs1, rs2, rd,

which performs an operation specified by the equation:

$$rd = (rs1*[rs1+1]) + rs2.$$

The term [rs1 + 1] designates data contained within the register following the explicitly defined register rs1. So, for example, if rs1 is set to specify a register r2, rs2 is set to specify a register r19, and rd is set to specify a register r22, then the instruction is as follows:

muladd r2, r19, r22,

and will perform the operation specified by the equation:

rd = (r2 * r3) + r19.

A register file 600 for a VLIW processor 100 (Figure 1) that supports implicitly-derived register specifiers is also discussed with reference to Figure 6. The processor has a decoder 602 that decodes instructions for execution in one or more of the functional units 620, 622, 624, and 626. The illustrated processor includes a multiported register file 600 that is divided into a plurality of separate register file segments 610, 612, 614, and 616. Each of the register file segments is associated with one of the plurality of functional units.

In some embodiments, decoder 602 decodes instructions that use implicitly-derived register specifiers and reads the explicitly-defined register. The decoder 602 then generates pointers both to the explicitly-defined register and to the implicitly-derived register. In other embodiments, a pointer to registers within the register file segments 610, 612, 614, and 616 includes an additional bit indicating that a register read is accompanied by a read of an implicitly-derived register.

ISSUE

No prima facie case of obviousness exists, because neither Baxter nor Tanenbaum, taken alone or in combination, discloses or suggests, expressly or inherently, executing an instruction that ***itself*** operates on both at least one register explicitly defined by a register specifier of the instruction and another register implicitly identified by the register specifier.

GROUPING OF CLAIMS

Group I: 1, 3-17, 19-21, and 23-24

ARGUMENTS

Arguments Pertaining to the Substance of the Art Rejections

Claim 1 requires a functional unit that executes an instruction that operates upon plural registers. At least one of these registers is explicitly identified by an explicitly defined register specifier. At least one other register is implicitly identified by the explicitly-defined register specifier. Claim 20 requires both explicitly defining a register specifier of a register operated upon during execution of an instruction, and implicitly deriving a register specifier of at least one other register operated on during execution of the instruction. Unfortunately, the PTO has improperly equated the limitations cited by the applicants with common autoindexing disclosed by Tanenbaum. As a result, the prior art has been accorded inordinate scope, the claims have been interpreted in a way that is overbroad, or both. Careful analysis will show that autoindexing, i.e. incrementing or decrementing an index register for use by a subsequently executed instruction, relies on a sequential-instruction type principle that is contrary to the Applicants' claimed invention. In this regard, the PTO has discounted important aspects of the claimed processor and method, particularly the operation of an instruction on both a register explicitly identified by an explicit register specifier and a register implicitly identified by the explicit register specifier. The nature of this legal error is now summarized.

Obviousness Rejections Under U.S.C. § 103

Claims 1, 2-17, 19-21, 23 and 24 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,826,096 (hereinafter, Baxter) in view of "Structured Computer Organization," by Andrew S. Tanenbaum (Prentice-Hall 1976) (hereinafter, Tanenbaum).

The legal standard for obviousness is defined in the Patent Statute, 35 U.S.C. § 103, which specifies, in addition to novelty requirements under § 102, further conditions for patentability relating to nonobvious subject matter. Those further conditions include the following:

> [a] patent may not be obtained though the invention is not identically disclosed or described [by prior art under 35 U.S.C. § 102] if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a

> whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

35 U.S.C. § 103 (1999).

Obviousness is a legal determination based on underlying factual inquiries. *Minnesota Min. & Mfg. Co. v. Johnson & Johnson Orthopaedics, Inc.*, 24 USPQ2d 1321, 1332-1333 (Fed. Cir. 1992). *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966) defines the factual inquiries utilized to evaluate the prior art. Specifically, the prior art is evaluated in terms of: (1) its scope and content; (2) the differences between the prior art and the claimed invention; (3) the level of ordinary skill in the art at the time the application was filed; and (4) objective, or secondary, evidence of nonobviousness such as commercial success, failure of others, long-felt need and unexpected results, which must be considered in reaching a conclusion of obviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17, 148 U.S.P.Q. 459, 460 (1966); *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1566-67, 1 U.S.P.Q.2d 1593, 1595-96 (Fed. Cir. 1987); *Minnesota Min. & Mfg. Co. v. Johnson & Johnson Orthopaedics, Inc.*, 24 U.S.P.Q.2d 1321, 1333 (Fed. Cir. 1992). In the present appeal, the pertinent issue involves the absence of certain features of the appealed claims in the relied upon references, resulting in a failure to make out a prima facie case of obviousness.

Obviousness analysis begins with a key legal question—what is the invention claimed? In this regard, the claimed invention must be evaluated as a whole. 35 U.S.C. § 103; *see also Panduit Corp.*, 1 U.S.P.Q.2d at 1597. Fundamentally, all claim limitations must be considered in the obviousness analysis. Indeed, it is clear error to ignore limitations clearly set forth in the claims. *Panduit Corp.*, 1 U.S.P.Q.2d at 1604. In general, multiple prior art references may be combined to provide a basis for an obviousness determination; however, there must be some teaching or suggestion for the combination. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1456 (Fed. Cir. 1998). Finally, a prior art reference must be considered in its entirety, *i.e.*, as a *whole*, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 220 U.S.P.Q. 303, 311 (Fed. Cir. 1983). Indeed, it is impermissible within the framework of § 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill in the art. *Bausch & Lomb, Inc. v. Barnes-Hind,*

Inc., 230 U.S.P.Q. 416 (Fed. Cir. 1986); *see also In re Wright*, 9 U.S.P.Q.2d 1649, 1652 (Fed. Cir. 1989).

No prima facie case of obviousness exists, because neither Baxter nor Tanenbaum discloses or suggests executing an instruction that operates on both at least one register explicitly defined by a register specifier of the instruction and another register implicitly identified by the register specifier.

GROUP I (Independent Claim 1)

Claims 1, 3-17, 19-21, and 23-24 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,826,096 (hereinafter, Baxter) in view of "Structured Computer Organization," by Andrew S. Tanenbaum (Prentice-Hall 1976) (hereinafter, Tanenbaum).

Independent claim 1, in pertinent part, requires:

...a functional unit...that executes an instruction that operates upon plural registers of said register file, including at least one register explicitly identified by an explicitly defined register specifier and at least one other register implicitly identified by the explicitly-defined register specifier.

The office action dated February 28, 2003, incorporated by reference into the most recent final action dated June 5, 2003, recites, "Baxter does not show how [a] register specifier is developed." The same office action recites, "If the operands of Baxter are consecutively stored, it would have been obvious to a person of ordinary skill in the art to use auto-indexing such that consecutively stored operands can be retrieved and acted upon by the opcodes."

The final office action dated June 5, 2003, recites, "Applicants contend that the applied references do not teach an instruction that operates upon plural registers (the wherein clause merely further describes in detail what an instruction operates upon plural registers means)." This statement indicates a misconception of the difference between the applicants' claimed invention and auto-indexing as disclosed in Tanenbaum.

In auto-indexing, a register is incremented or decremented, *in preparation for use by a subsequent instruction*. Auto indexing is, therefore, inconsistent with the applicant's claimed invention, which requires implicitly identifying an additional register from an explicit register specifier of an instruction, where *the instruction operates on both the implicitly and explicitly identified registers*.

As illustrated above, neither Baxter nor Tanenbaum, taken individually or in combination, discloses or suggests, either explicitly or implicitly, a processor including "a functional unit...that executes an instruction that operates upon plural registers of said register file, including at least one register explicitly identified by an explicitly defined register specifier and at least one other register implicitly identified by the explicitly-defined register specifier," as required by independent claim 1.

CONCLUSION

For the at least the foregoing reasons, Appellants' presently claimed invention would not have been obvious to one of ordinary skill in the art under 35 U.S.C. § 103(a) in view of the cited prior art. Accordingly, this honorable Board is respectfully requested to reverse the rejections of Claims 1, 3-17, 19-21, and 23-24 and to direct the claims of the present application to be issued.

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that, on the date shown below, this correspondence is being

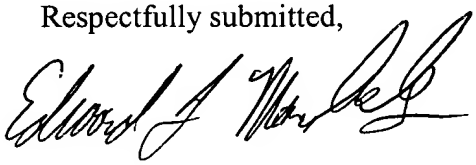
☐ deposited with the US Postal Service with sufficient postage as first class mail, in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

☐ facsimile transmitted to the US Patent and Trademark Office.

Edward J. Marshall _____ Date _____

EXPRESS MAIL LABEL: EL989617357US

Respectfully submitted,



Edward J. Marshall, Reg. No. 45,395
Attorney for Applicant(s)
(512) 338-6321
(512) 338-6301 (fax)

APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

1. A processor comprising:

a register file; and

a functional unit, coupled to the register file, that executes an instruction that operates upon plural registers of said register file, including at least one register explicitly identified by an explicitly defined register specifier and at least one other register implicitly identified by the explicitly-defined register specifier.

3. A processor according to Claim 1 wherein:

a register specifier for the other register is implicitly derived by adding one to the explicitly-defined register specifier.

4. A processor according to Claim 1 wherein the processor is a Very Long Instruction Word (VLIW) processor and wherein

said register file further including a plurality of register file segments wherein the plurality of registers are divided among said plurality of register file segments; and

wherein said VLIW processor further comprises a plurality of functional units, ones of the plurality of functional units being coupled to and associated with respective ones of the register file segments.

5. A processor according to Claim 1 wherein:

the instruction is a multiply-add instruction that uses an implicitly-derived register specifier and has a form of:

muladd rsl, rs2, rd,

and performs an operation specified by the equation:

$$rd = (rsl * [rsl+1]) + rs2,$$

where the term [rsl+ 1] designates data contained within the other register following the explicitly-defined register rsl.

6. A processor according to Claim 1 wherein:

the instruction is a bit extract instruction that uses an implicitly-derived register specifiers and has a form of:

bitext rsl, rs2, rd,

and performs an operation of extracting bits from even-aligned pairs of registers r[rsl] and [rsl+ 1] where the term [rsl+l] designates data contained within the other register following the explicitly-defined register rsl, wherein data in register r[rs2] describes the extracted field of registers r[rsl] and r[rsl + 1] and register r[rd] is a destination register.

7. A processor according to Claim 1 wherein:

the instruction is a call instruction that uses an implicitly-derived register specifiers and has a form of:

call label,

causing a control transfer to an address specified by a label operand, the address of the instruction word following the instruction word begun with the call instruction is held in an alias register, an implicit operand of the call instruction, with an assembler using an alias link pointer lp for pointing to the alias register.

8. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point add instruction that uses an implicitly-derived register specifiers and has a form of:

dadd rsl, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] =(rsl, [rsl+1])+(rs2, [rs2+1]),

where the terms (rsl, [rsl+1]), (rs2, [rs2+1]), and (rd, [rd+l]) designate double-precision words.

9. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point compare instruction that uses an implicitly-derived register specifiers and has a form of:

dcmpcc rsl, rs2, rd,

and performs an operation of comparing data in registers (rsl, [rsl+ 1]) with data in registers (rs2, [rs2+1]) and storing a result in registers (rd,[rd+l]) where the terms (rsl, [rsl+l]), (rs2, [rs2+ 1]), and (rd, [rd+ 1]) designate double-precision words, and cc designates a condition code including equal, less than, and less than or equal to conditions.

10. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point multiply instruction that uses an implicitly-derived register specifiers and has a form of:

droul rs1, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] =(rsl, [rsl+1]) * (rs2, [rs2+1]),

where the terms (rsl, [rsl+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

11. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point subtraction instruction that uses an implicitly-derived register specifiers and has a form of:

dsub rsl, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+l]) = (rsl, [rsl+l]) - (rs2, [rs2+1]),

where the terms (rsl, [rsl+l]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

12. A processor according to Claim 1 wherein:

the instruction is a pack instruction that uses an implicitly-derived register specifiers and has a form of:

pack rsl, rs2, rd,

and operates upon a register pair (rsl, [rsl+l]) as four signed 16-bit operands, and shifts the four operands right by a value designated by the register specified by rs2, clips the shifted 16-bit operands within defined limits and stores the clipped 16-bit operands in a register pair (rd, [rd+1]).

13. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point conversion instruction that uses an implicitly-derived register specifiers and has a form of:

dtox rsl, rd,

and performs an operation of converting a double-precision floating point value to a specified format x, the format x including a single-precision floating point format (dtof), an integer format (dtoi), and a long integer format (dtol).

14. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point absolute value instruction that uses an implicitly-derived register specifiers and has a form of:

dabs rsl, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rsl, [rsl+ 1]) to an absolute magnitude in a register pair (rd, [rd+ 1]).

15. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point negative value instruction that uses an implicitly-derived register specifiers and has a form of:

dneg rs1, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rs1, [rs1 +1]) to a negative magnitude in a register pair (rd, [rd+1]).

16. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point set limit instruction that uses an implicitly-derived register specifiers and has a form of:

dlim rsl, rs2, rd,

and performs an operation of setting a double-precision destination register (rd, [rd+ 1]) to the maximum of a double-precision first source register (rsl, [rsl+1]) and a double- precision second source register (rs2, [rs2+ 1]), or setting the double-precision destination register (rd, [rd+l]) to the minimum of a double precision first source register (rsl, [rsl+1]) and a double precision second source register (rs2, [rs2+ 1]),

where the terms (rsl, [rsl+1]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double- precision words.

17. A processor according to Claim 1 further comprising:

a decoder coupled to the functional unit and configured to generate a first pointer pointing to the explicitly-specified register and a second pointer pointing to the other register.

19. A processor according to Claim 1 further comprising:

a pointer coupled to the register file and designating a register in the register file, the pointer including a signal indicative of selection of a the other register, wherein a register read of the explicitly-specified register is accompanied by a register read of the other register when implicit derivation of the explicitly-defined register specifier is selected.

20. A method of operating a processor comprising:

storing information in a register file including a plurality of registers;

executing instructions in a functional unit coupled to the register file and operating upon a plurality of registers in the register file;

explicitly defining a register specifier of a register operated upon during executing of the instruction; and

implicitly deriving a register specifier of at least one other register operated upon during executing of the instruction based on the explicitly defined register specifier.

21. A method according to Claim 20 further comprising:

decoding an instruction; and

deriving, during decoding of the instruction, a register specifier based on the explicitly-specified register specifier of the instruction.

23. The processor of claim 1, wherein the register specifier is encoded as an indirect specifier.

24. A method according to Claim 20 further comprising:

implicitly deriving the register specifier for the other register by adding one to the explicitly-defined register specifier.